ISIMA 1ère année

22 janvier 2016
Durée : 2 heures

PROGRAMMATION FONCTIONNELLE

Exercice :

Une liste de nombres $(p_1, \ldots, p_n)$ constitue une distribution antiuniforme si et seulement si les deux conditions suivantes sont satisfaites :

1°) $\sum_{i=1}^{n} p_i = 1$;

2°) $n \leq 3$ ou ($n > 3$ et $\forall\, i \in [\![1, n-3]\!], p_i \geq p_{i+2} + p_{i+3} + \ldots + p_n$).

Ecrire une fonction Scheme `antiUnif` ayant comme arguments une liste de nombres `L` et telle que l'évaluation de l'expression `(antiUnif L)` retourne `#t` si la liste `L` représente une distribution antiuniforme et `#f` sinon.

Problème :

Rappel : *On appelle schéma d'application itératif les schémas de type* `map` *ou* `append-map`.

Le but de ce problème est d'écrire, en Scheme, une fonction ayant comme argument une famille de k vecteurs $(V_1, ..., V_k)$ de $\mathbb{R}^n$ et qui retourne une famille de k vecteurs $(W_1, ..., W_k)$ de $\mathbb{R}^n$ telle que l'espace engendré par $(W_1, ..., W_k)$ (*i.e.* l'ensemble de tous les vecteurs que l'on peut obtenir par combinaison linéaire de $W_1, ..., W_k$ soit le même que celui engendré par $V_1, ..., V_k$, et telle que les vecteurs $W_1, ..., W_k$ soient orthogonaux entre eux et de norme égale à 1 (famille orthonormée).

Un vecteur de $\mathbb{R}^n$ peut être représenté par une liste de n nombres du type $(x_1 \ldots x_n)$.

1°) Ecrire, en utilisant un schéma d'application itératif, une fonction `AddVect` ayant comme arguments deux vecteurs `X` et `Y` respectivement du type $(x_1 \ldots x_n)$ et $(y_1 \ldots y_n)$ et telle que l'évaluation de l'expression `(AddVect X Y)` retourne l'addition de `X` et `Y`, c'est-à-dire un vecteur du type $(x_1 + y_1 \ldots x_n + y_n)$.

2°) Ecrire, en utilisant un schéma d'application itératif, une fonction `MultExt` ayant comme arguments un nombre réel `k` et un vecteur `X` du type $(x_1 \ldots x_n)$ et telle que l'évaluation de l'expression `(MultExt k X)` retourne la multiplication externe de `k` par `X`, c'est-à-dire un vecteur du type (k x_1 ... k x_n).

3°) Ecrire, en utilisant un schéma d'application itératif, une fonction `Prodscal` ayant comme arguments deux vecteurs `X` et `Y` respectivement du type $(x_1 \ldots x_n)$ et $(y_1 \ldots y_n)$ et telle que

l'évaluation de l'expression `(Prodscal X Y)` retourne le produit scalaire de `X` et `Y`, c'est-à-dire un nombre réel qui s'exprime sous la forme $\sum_{i=1}^{n} x_i y_i$.

4°) Ecrire une fonction `Norme` ayant comme argument un vecteur `X` du type $(x_1 \ldots x_n)$ et telle que l'évaluation de l'expression `(Norme X)` retourne la norme euclidienne de `X`, c'est-à-dire un nombre réel qui s'exprime sous la forme $\sqrt{\sum_{i=1}^{n} x_i^2}$.

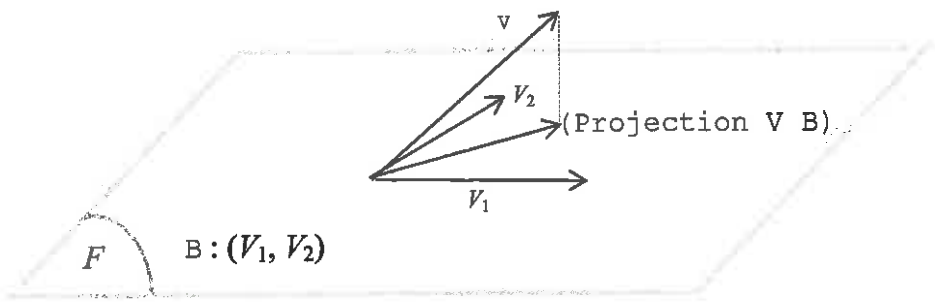
On pourra utiliser la fonction prédéfinie `sqrt` *qui retourne la racine carrée de son argument.*

5°) Ecrire une fonction `SomListVect` ayant comme argument une liste `LV` de vecteurs du type $(V_1 \ldots V_n)$ et telle que l'évaluation de l'expression `(SomListVect LV)` retourne la somme de tous les vecteurs de la liste `LV`, c'est-à-dire un vecteur du type $V_1 + \ldots + V_n$, où le symbole "+" désigne ici l'addition vectorielle (`AddVect`).

6°) Si on note F l'espace engendré par n vecteurs $V_1, \ldots, V_n$, on appelle projeté orthogonal d'un vecteur X sur F le vecteur Y défini par $Y = \sum_{i=1}^{n} (X, V_i) \cdot V_i$, où la notation (X,V_i) désigne le produit scalaire entre X et V_i (`Prodscal`), le symbole "." désigne la multiplication externe (`MultExt`) et le symbole "Σ" l'addition vectorielle.

Ecrire, en utilisant un schéma d'application itératif, une fonction `Projection` ayant comme arguments un vecteur `V` et une liste de vecteurs `B` du type $(V_1 \ldots V_n)$ et telle que l'évaluation de l'expression `(Projection V B)` retourne le projeté orthogonal de `V` sur l'espace engendré par les vecteurs de `B`.

Exemple :



7°) Pour tout entier i, on note H_i l'espace engendré par $(V_1, ..., V_i)$ et p_i la projection orthogonale sur H_i.

Un algorithme pour construire une famille orthonormée $(W_1, ..., W_k)$ qui engendre le même espace que $(V_1, ..., V_k)$ est le suivant :

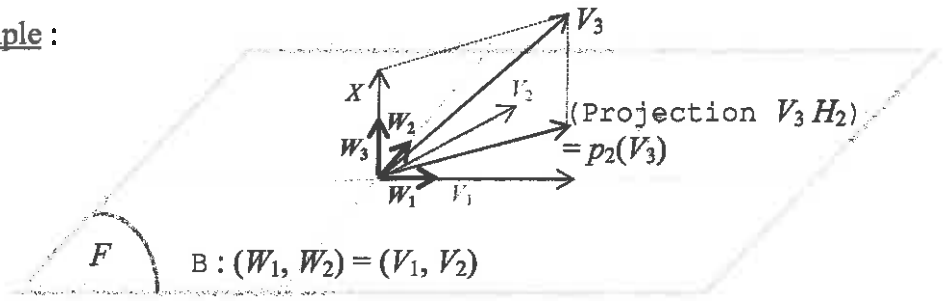
1°) $W_1 \leftarrow V_1 / \| V_1 \|$;

2°) Pour i variant de 2 à n faire

$X \leftarrow V_i - p_{i-1}(V_i)$; // p_{i-1} est la projection orthogonale sur H_{i-1}

$W_i \leftarrow X / \| X \|$.

Exemple :



Ecrire une fonction `Ortho` ayant comme argument une liste `LV` de vecteurs du type $(V_1 \ldots V_k)$ et telle que l'évaluation de l'expression `(Ortho LV)` retourne une liste de vecteurs du type $(W_1, ..., W_k)$ telle que les vecteurs $W_1, ..., W_k$ soient orthogonaux entre eux, de norme égale à 1 et tels que l'ensemble de tous les vecteurs que l'on peut obtenir par combinaison linéaire de $W_1, ..., W_k$ soit le même que celui engendré par $V_1, ..., V_k$.

Remarque : l'ordre dans lequel on traite les vecteurs n'a pas d'importance.